



Ulrike Meier Yang
Center for Applied Scientific Computing

ACTS Toolkit Workshop
October 10-13, 2001



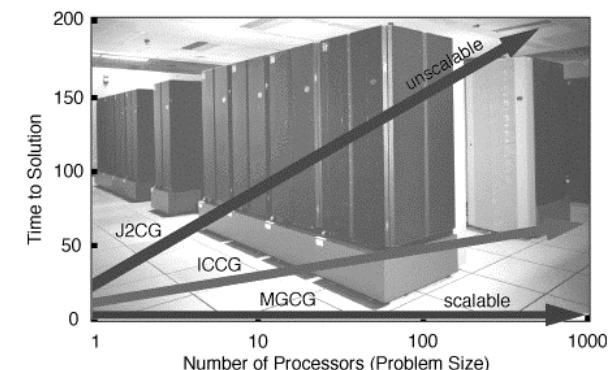
Outline

- Introduction / Motivation
- Getting Started / Conceptual Interfaces
- Structured-Grid Interface (**struct**)
- Semi-Structured-Grid Interface (**sstruct**)
- Finite Element Interface (**FEI**)
- Linear-Algebraic Interface (**IJ**)
- Solvers and Preconditioners
- Additional Information

Hypre team

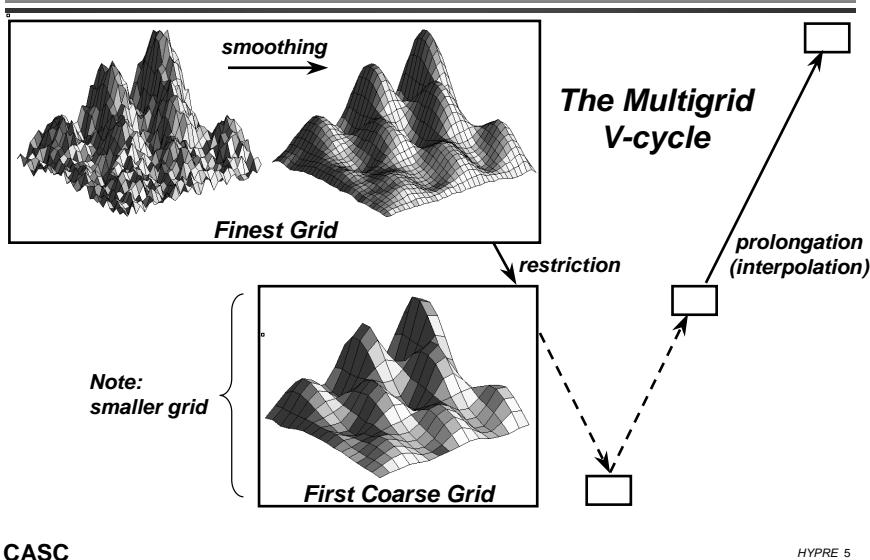
- Andy Cleary (project leader)
- Edmond Chow
- Rob Falgout
- Van Emden Henson
- Jim Jones
- Mike Lambert
- Jeff Painter
- Charles Tong
- Tom Treadway
- Ulrike Meier Yang

Scalability is a central issue for large-scale parallel computing



Linear solver convergence can be discussed independent of parallel computing, and is often overlooked as a key scalability issue.

Multigrid uses coarse grids to efficiently damp out smooth error components



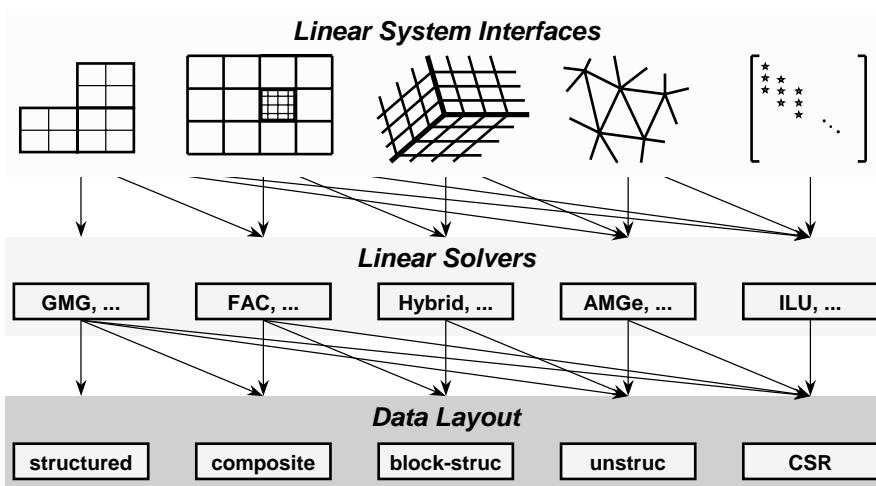
Getting Started

- Before writing your code:
 - choose a conceptual interface
 - choose a solver / preconditioner
 - choose a matrix type that is compatible with your solver / preconditioner and conceptual interface
- Now write your code:
 - build auxiliary structures (e.g., grids, stencils)
 - build matrix/vector through conceptual interface
 - build solver/preconditioner
 - solve the system
 - get desired information from the solver

CASC

HYPRE 6

Multiple interfaces are necessary to provide “best” solvers and data layouts



Why multiple interfaces? The key points

- Provides natural “views” of the linear system
- Eases some of the coding burden for users by eliminating the need to map to rows/columns
- Provides for more efficient (scalable) linear solvers
- Provides for more effective data storage schemes and more efficient computational kernels

CASC

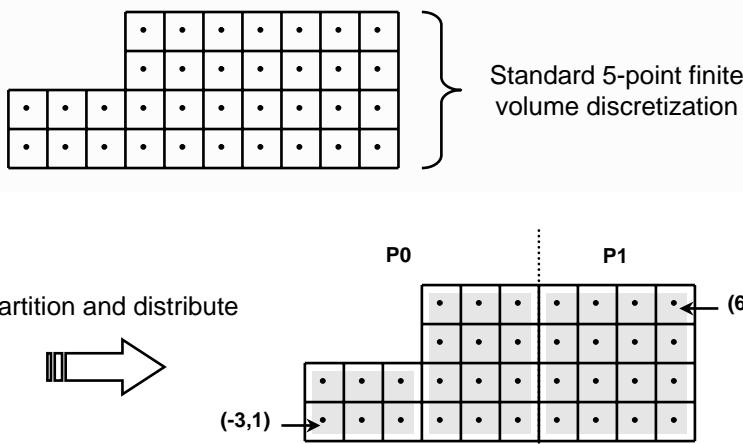
HYPRE 8

hypre currently supports four conceptual interfaces

- **Structured-Grid Interface (struct)**
 - applications with logically rectangular grids
- **Semi-Structured-Grid Interface (sstruct)**
 - applications with grids that are mostly structured, but with some unstructured features (e.g., block-structured, AMR, overset)
- **Finite Element Interface (FEI)**
 - unstructured-grid, finite element applications
- **Linear-Algebraic Interface (IJ)**
 - applications with sparse linear systems
- More about each next...

CASC

HYPRE 9



CASC

HYPRE 11

Structured-Grid System Interface (Struct)

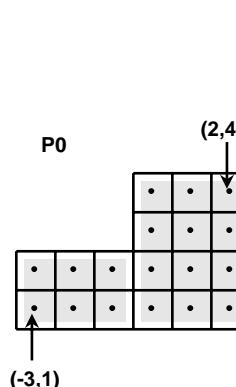
- There are four basic steps involved:
 - set up the Grid
 - set up the Stencil
 - set up the Matrix
 - set up the right-hand-side Vector
- Consider the following 2D Laplacian problem

$$\begin{cases} \nabla^2 u = f, & \text{in the domain} \\ u = 0, & \text{on the boundary} \end{cases}$$

CASC

HYPRE 10

A structured-grid finite volume example : setting up the Grid



```
HYPRE_StructGrid grid;
int ndim = 2;
int ilo[2][2] = {{-3, 1}, {0, 1}};
int iup[2][2] = {{-1, 2}, {2, 4}};

HYPRE_StructGridCreate(MPI_COMM_WORLD,
                      ndim, &grid);

HYPRE_StructGridSetExtents(grid,
                           ilo[0], iup[0]);
HYPRE_StructGridSetExtents(grid,
                           ilo[1], iup[1]);

HYPRE_StructGridAssemble(grid);
```

CASC

HYPRE 12

A structured-grid finite volume example : setting up the Stencil

$$\begin{pmatrix} (0,1) \\ (-1,0) (0,0) (1,0) \\ (0,-1) \end{pmatrix}$$



$$\begin{pmatrix} S4 \\ S1 S0 S2 \\ S3 \end{pmatrix}$$

CASC

```
HYPRE_StructStencil stencil;
int ndim = 2;
int size = 5;
int offsets[5][2] = {{0,0}, {-1,0}, {1,0}, {0,-1}, {0,1}};
int s;

HYPRE_StructStencilCreate(ndim, size,
    &stencil);

for (s = 0; s < 5; s++)
{
    HYPRE_StructStencilSetElement(
        stencil, s, offsets[s]);
}
```

HYPRE 13

A structured-grid finite volume example : setting up the Matrix

$$\begin{array}{c} P0 \\ \uparrow \downarrow \\ \begin{matrix} & & & (2,4) \\ & & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \end{matrix} \\ (-3,1) \end{array}$$

$$\begin{pmatrix} S4 \\ S1 S0 S2 \\ S3 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 4 -1 \\ -1 \end{pmatrix}$$

CASC

```
HYPRE_StructMatrix A;
double values[36] = {4, -1, 4, -1, ...};
int nentries = 2;
int entries[2] = {0,3};

HYPRE_StructMatrixCreate(MPI_COMM_WORLD,
    grid, stencil, &A);
HYPRE_StructMatrixInitialize(A);

HYPRE_StructMatrixSetBoxValues(A,
    ilo[0], iup[0], nentries, entries,
    values);
HYPRE_StructMatrixSetBoxValues(A,
    ilo[1], iup[1], nentries, entries,
    values);

/* set boundary conditions */
...
HYPRE_StructMatrixAssemble(A);
```

HYPRE 14

A structured-grid finite volume example : setting up the Matrix (bc's)

$$\begin{array}{c} P0 \\ \uparrow \downarrow \\ \begin{matrix} & & & (2,4) \\ & & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \end{matrix} \\ (-3,1) \quad (2,1) \end{array}$$

$$\begin{pmatrix} S4 \\ S1 S0 S2 \\ S3 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 4 -1 \\ 0 \end{pmatrix}$$

CASC

```
int ilo[2] = {-3, 1};
int iup[2] = { 2, 1};
double values[12] = {0, 0, ...};
int nentries = 1;

/* set interior coefficients */
...
/* implement boundary conditions */
...
i = 3;
HYPRE_StructMatrixSetBoxValues(A,
    ilo, iup, nentries, &i, values);

/* complete implementation of bc's */
...
```

HYPRE 15

A structured-grid finite volume example : setting up the right-hand-side Vector

$$\begin{array}{c} P0 \\ \uparrow \downarrow \\ \begin{matrix} & & & (2,4) \\ & & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \end{matrix} \\ (-3,1) \end{array}$$

```
HYPRE_StructVector b;
double values[18] = {0, 0, ...};

HYPRE_StructVectorCreate(MPI_COMM_WORLD,
    grid, &b);
HYPRE_StructVectorInitialize(b);

HYPRE_StructVectorSetBoxValues(b,
    ilo[0], iup[0], values);
HYPRE_StructVectorSetBoxValues(b,
    ilo[1], iup[1], values);

HYPRE_StructVectorAssemble(b);
```

CASC

HYPRE 16

Symmetric Matrices

- Some solvers support symmetric storage
- Between Create() and Initialize(), call:

```
HYPRE_StructMatrixSetSymmetric(A, 1);
```

- For best efficiency, only set half of the coefficients

$$\begin{pmatrix} (0,1) \\ (0,0) (1,0) \end{pmatrix} \leftrightarrow \begin{pmatrix} s_2 \\ s_0 \quad s_1 \end{pmatrix}$$

- This is enough info to recover the full 5-pt stencil

CASC

HYPRE 17

Semi-Structured-Grid System Interface (SStruct)

- There are five basic steps involved:
 - set up the Grid
 - set up the Stencils
 - set up the Graph
 - set up the Matrix
 - set up the right-hand-side Vector
- Consider again the following 2D Laplacian problem

$$\begin{cases} \nabla^2 u = f, & \text{in the domain} \\ u = 0, & \text{on the boundary} \end{cases}$$

CASC

HYPRE 19

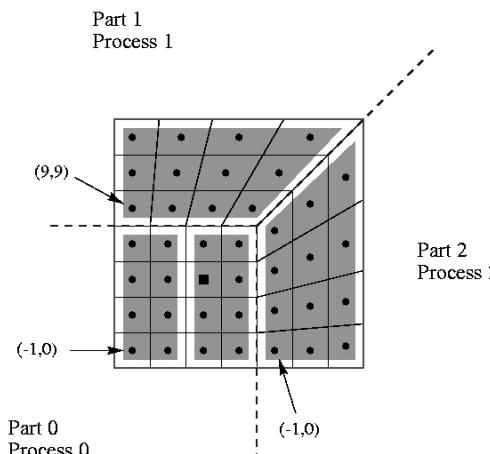
Semi-Structured-Grid System Interface (SStruct)

- Allows more general grids
 - Grids that are mostly structured but have some unstructured features
 - Examples: Structured adaptive mesh refinement, Block-structured grids, Overset grids
- Allows more general PDE's
 - Multiple variables (system PDE's)
 - Multiple variable types (cell centered, face centered, vertex centered, ...)

CASC

HYPRE 18

A block-structured grid example



CASC

HYPRE 20

A block-structured grid example : setting up the Grid

```

    | HYPRE_SStructVariable vars[1] =
    |     {HYPRE_SSTRUCT_VARIABLE_CELL};
    |     int addindex[2] = {1,2};
    |     HYPRE_SStructVariable addvars[1] =
    |         {HYPRE_SSTRUCT_VARIABLE_CELL};
    |     int nparts = 3, part = 0;
    |     int nvars = 1, naddvars = 1;

    |     HYPRE_SStructGridCreate(MPI_COMM_WORLD,
    |         ndim, nparts, &grid);
    |     HYPRE_SStructGridSetExtents(grid, part,
    |         ilo[0], iup[0]);
    |     ...
    |     HYPRE_SStructGridSetVariables(grid, part,
    |         nvars, vars);
    |     HYPRE_SStructGridAddVariables(grid, part,
    |         addindex, naddvars, addvars);

    |     HYPRE_SStructGridAssemble(grid);

```

HYPRE 21

CASC

A block-structured grid example : setting up the Stencil

```

    HYPRE_SStructStencil stencil;
    int ndim = 2, size = 9, var = 0;
    int s;
    int offsets[9][2] = {{0,0},
                         {-1, 0}, {1, 0},
                         {0,-1}, {0, 1},
                         {-1,-1}, {1,-1},
                         {-1, 1}, {1, 1}};

    HYPRE_SStructStencilCreate(ndim, size,
                               &stencil);

    for (s = 0; s < 9; s++)
    {
        HYPRE_SStructStencilSetEntry(stencil,
                                     s, offsets[s], var);
    }

```

HYPRE 22

CASC

A block-structured grid example : setting up the Graph

```

    HYPRE_SStructGraph graph;
    int part = 0, var = 0;
    int to_part = 2, to_var = 0;
    int index[2] = {2,0};
    int to_index[2] = {-1,0};

    HYPRE_SStructGraphCreate(MPI_COMM_WORLD,
                           grid, &graph);
    HYPRE_SStructGraphSetStencil(graph,
                                part, var, stencil);

    /* Add entries at part boundaries */

    HYPRE_SStructGraphAddEntries(graph,
                                part, index, var,
                                to_part, to_index, to_var);
    ...

    HYPRE_SStructGraphAssemble(graph);

```

HYPRE 23

CASC

A block-structured grid example : setting up the Matrix

```

    int sentries[2] = {0,3}, nsentries = 2;
    int gentries[1] = {9}, ngentries = 1;

    HYPRE_SStructMatrixCreate(MPI_COMM_WORLD,
                           graph, &A);
    HYPRE_SStructMatrixInitialize(A);
    HYPRE_SStructMatrixSetBoxValues(A,
                                    part, ilo[0], iup[0], var,
                                    nsentries, sentries, values);
    ...

    /* set values at non-stencil entries */
    HYPRE_SStructMatrixSetValues(A,
                                part, index[0], var, ngentries,
                                gentries, values);
    ...

    /* zero entries outside domain/part */
    /* set bc's at domain boundaries, */

    HYPRE_SStructMatrixAssemble(A);

```

HYPRE 24

CASC

Building different matrix/vector storage formats with the SStruct interface

- Efficient preconditioners often require specific matrix/vector storage schemes
- Between `Create()` and `Initialize()`, call:

```
HYPRE_SStructMatrixSetObjectType(A, HYPRE_PARCSR);
```

- After `Assemble()`, call:

```
HYPRE_SStructMatrixGetObject(A, &parcsr_A);
```

- Now, use the ParCSR matrix with compatible solvers such as BoomerAMG (algebraic multigrid)

CASC

HYPRE 25

Linear-Algebraic System Interface (IJ)

- The IJ interface provides access to general sparse-matrix solvers, but not specialized solvers
- There are two basic steps involved:
 - set up the Matrix
 - set up the right-hand-side Vector
- Consider a 5pt 2D Laplacian problem on a 10x10 grid

$$\begin{pmatrix} A & -I & & \\ -I & A & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & A \end{pmatrix}, \quad A = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & \ddots & & \\ & \ddots & \ddots & \ddots & -1 \\ & & -1 & 4 & \end{pmatrix}$$

CASC

HYPRE 27

Finite Element Interface (FEI)

- The FEI interface is designed for finite element discretizations on unstructured grids
- See the following for information on how to use it

R. L. Clay et al. An annotated reference guide to the Finite Element Interface (FEI) Specification, Version 1.0. Sandia National Laboratories, Livermore, CA, Technical Report SAND99-8229, 1999.

CASC

HYPRE 26

An example for the IJ interface: setting up the Matrix

```
HYPRE_IJMatrix A;
HYPRE_ParCSRMatrix parcsr_A;
int nrows = 3, ncols[3] = {3, 4, 4}, rows[3] = {0, 1, 2};
int cols[[11] = {0,1,10,0,1,2,11,1,2,3,12};
double values[11] = {4,-1,-1, -1, 4, -1, -1, -1, 4, -1,-1};

HYPRE_IJMatrixCreate(MPI_COMM_WORLD, ilower, iupper,
                     jlower, jupper, &A);
HYPRE_IJMatrixSetObjectType(A, HYPRE_PARCSR);

HYPRE_IJMatrixInitialize(A);

/* set matrix coefficients several rows at a time */
HYPRE_IJMatrixSetValues(A, nrows, ncols,
                        rows, cols, values);
...
HYPRE_IJMatrixAssemble(A);
HYPRE_IJMatrixGetObject(A, &parcsr_A);
```

CASC

HYPRE 28

An example for the IJ interface: setting up the right-hand-side Vector

```
HYPRE_IJVector b;
HYPRE_ParVector par_b;
int jlower, jupper, nvalues = 100;
int indices[100] = {0,1,...,99};
double values[100] = {1,1,...,1};

HYPRE_IJVectorCreate(MPI_COMM_WORLD, jlower, jupper, &b);
HYPRE_IJVectorSetObjectType(b, HYPRE_PARCSR);

HYPRE_IJVectorInitialize(b);

/* set several coefficients at a time */
HYPRE_IJVectorSetValues(b, nvalues, indices, values);
...
HYPRE_IJVectorAssemble(b);
HYPRE_IJVectorGetObject(b, &par_b);
```

CASC

HYPRE 29

Additional IJ interface function calls

IJMatrix function calls:

```
HYPRE_IJMatrixAddToValues(A, nrows, ncols,
                           rows, cols, values)

HYPRE_IJMatrixGetValues(A, nrows, ncols, rows,
                        cols, values)
```

For better efficiency:

```
HYPRE_IJMatrixSetRowSizes(A, sizes)
HYPRE_IJMatrixSetDiagOffdSizes(A, diag_sizes,
                               offdiag_sizes)
```

IJVector function calls:

```
HYPRE_IJVectorAddToValues(b, nvalues, indices,
                           CASCvalues)
```

HYPRE 30

Several Solvers and Preconditioners are available in *hypre*

- Current solver availability via conceptual interfaces

Solvers	Struct	SStruct	FEI	IJ
Jacobi	X			
SMG	X			
PFMG	X			
BoomerAMG	X	X	X	X
ParaSails	X	X	X	X
PILUT	X	X	X	X
Euclid	X	X	X	X
PCG	X	X	X	X
GMRES	X	X	X	X

CASC

HYPRE 31

Setup and use of solvers is largely the same (see Reference Manual for details)

• Create the solver

```
HYPRE_SolverCreate(MPI_COMM_WORLD,
&solver);
```

• Set parameters

```
HYPRE_SolverSetTol(solver, 1.0e-06);
```

• Prepare to solve the system

```
HYPRE_SolverSetup(solver, A, b, x);
```

• Solve the system

```
HYPRE_SolverSolve(solver, A, b, x);
```

• Get solution info out via conceptual interface

```
HYPRE_StructVectorGetValues(struct_x,
                             index, values);
```

• Destroy the solver

HYPRE 32

Solver example: SMG-PCG

```
HYPRE_StructPCGCreate(MPI_COMM_WORLD, &solver);

/* set stopping criteria */
HYPRE_StructPCGSetTol(solver, 1.0e-06);

/* define preconditioner (one symmetric V(1,1)-cycle) */
HYPRE_StructSMGCreate(MPI_COMM_WORLD, &precond);
HYPRE_StructSMGSetMaxIter(precond, 1);
HYPRE_StructSMGSetTol(precond, 0.0);
HYPRE_StructSMGSetZeroGuess(precond);
HYPRE_StructSMGSetNumPreRelax(precond, 1);
HYPRE_StructSMGSetNumPostRelax(precond, 1);

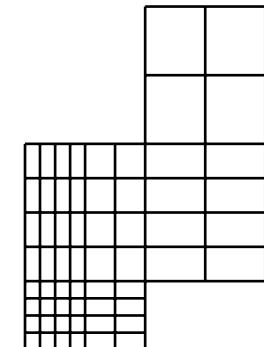
/* set preconditioner and solve */
HYPRE_StructPCGSetPrecond(solver,
    HYPRE_StructSMGSolve, HYPRE_StructSMGSetup, precond);
HYPRE_StructPCGSetup(solver, A, b, x);
HYPRE_StructPCGSolve(solver, A, b, x);
```

CASC

HYPRE 33

SMG and PFMG are semicoarsening multigrid methods for structured grids

- **Interface:** Struct
- **Matrix Class:** Struct
- **SMG** uses plane smoothing in 3D, where each plane “solve” is effected by one 2D V-cycle
- **SMG** is very robust
- **PFMG** uses simple pointwise smoothing, and is less robust



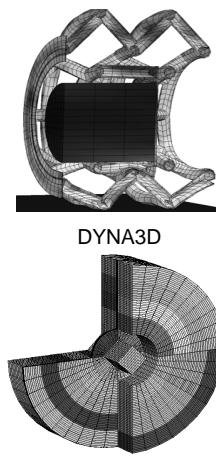
Our largest run (with PFMG-CG): 1B unknowns on 3150 processors of ASCI Red in 54 seconds

CASC

HYPRE 34

BoomerAMG is an algebraic multigrid method for unstructured grids

- **Interface:** SStruct, FEI, IJ
- **Matrix Class:** ParCSR
- Originally developed as a general matrix method (i.e., assumed given A, x, and b only)
- Uses simple pointwise relaxation
- Automatically defines coarse “grids”



CASC

HYPRE 35

BoomerAMG interface functions

Coarsening techniques:

- CLJP coarsening
- various variants of the classical Ruge-Stüben (RS) coarsening
- Falgout coarsening (default)

`HYPRE_BoomerAMGSetCoarsenType(solver, coarsen_type)`

Relaxation techniques:

- weighted Jacobi relaxation
- hybrid Gauß-Seidel / Jacobi relaxation (default)
- symmetric hybrid Gauß-Seidel / Jacobi relaxation
- further techniques underway

`HYPRE_BoomerAMGSetGridRelaxType(solver, relax_type)`

HYPRE 36

More BoomerAMG interface functions

```
HYPRE_BoomerAMGSetMaxLevels(solver,  
    max_levels)  
    default: 25  
  
HYPRE_BoomerAMGSetMaxIter(solver, max_iter)  
    default: 20  
  
HYPRE_BoomerAMGSetTol(solver, tol)  
    default:  
  
HYPRE_BoomerAMGSetStrongThreshold(solver,  
    strong_thr)  
    default: 0.25  
  
HYPRE_BoomerAMGSetMaxRowSum(solver,  
    max_row_sum)  
    for more efficient treatment of very diagonally dominant portions  
    of the matrix, default: 0.9
```

HYPRE 37

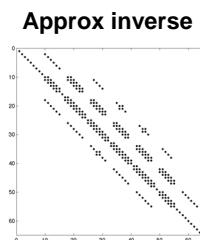
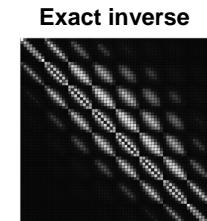
ParaSails interface functions

```
HYPRE_ParaSailsCreate(MPI_Comm comm,  
    HYPRE_Solver *solver, int symmetry)  
    0 nonsymm./indefinite problem, nonsymm.  
    Preconditioner (default)  
    1 SPD problem, SPD (factored) preconditioner  
    2 nonsymm./definite problem, SPD (factored)  
    preconditioner  
  
HYPRE_ParaSailsSetParams(HYPRE_Solver  
    solver, double thresh, int nlevel,  
    double filter)  
    thresh: drops all entries in symm. diagonally scaled  
    A less than thresh, default: 0.1  
    nlevel: m = nlevel+1 , default: 1  
    filter: post-thresholding procedure, default: 0.1  
  
CASC HYPRE_ParaSailsSetLoadbal(solver)
```

HYPRE 39

ParaSAILS is an approximate inverse method for sparse linear systems

- Interface: SStruct, FEI, IJ
- Matrix Class: ParCSR



CASC

HYPRE 38

PILUT is an Incomplete LU method for sparse linear systems

- Interface: SStruct, FEI, IJ
 - Matrix Class: ParCSR
-
- Uses thresholding drop strategy plus a mechanism to control the maximum size of the ILU factors
 - originally by Kumar and Karypis for T3D
 - now uses MPI and more coarse-grain parallelism
 - uses Schur-complement approach to parallelism

CASC

HYPRE 40

PILUT interface functions

```
HYPRE_ParCSRPIlutSetDropTolerance (solver, tol)
default: 0.0001
```

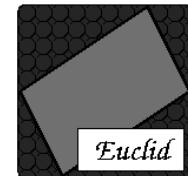
```
HYPRE_ParCSRPIlutSetFactorRowSize (solver, size)
default: 20
```

CASC

HYPRE 41

Euclid is a family of Incomplete LU methods for sparse linear systems

- **Interface:** SStruct, FEI, IJ
- **Matrix Class:** ParCSR



- Obtains scalable parallelism via local and global reorderings
 - Good for unstructured problems
-
- <http://www.cs.odu.edu/~hysom/Euclid>

CASC

HYPRE 42

Euclid interface functions

```
HYPRE_EuclidSetParams(solver, argc, argv)
if passed on command line
or
HYPRE_EuclidSetParam(solver, name, value)
for each parameter
```

Options:

- level factorization level for ILU(k)
- bj use block Jacobi preconditioning instead of PILU
- eu_stats prints summary of timing information
- eu_mem prints summary of memory usage

CASC

HYPRE 43

Krylov solvers interface functions

```
HYPRE_ParCSRPCGSetMaxIter(solver, max_iter)
HYPRE_ParCSRPCGSetTol(solver, tol)
HYPRE_ParCSRPCGGetNumIterations(solver, iter)
HYPRE_ParCSRPCGGetFinalRelativeResidualNorm(solver, norm)
```

```
HYPRE_ParCSRGMRESSetKDim(solver, k_dim)
HYPRE_ParCSRGMRESSetMaxIter(solver, max_iter)
HYPRE_ParCSRGMRESSetTol(solver, tol)
HYPRE_ParCSRGMRESGetNumIterations(solver, iter)
HYPRE_ParCSRGMRESGetFinalRelativeResidualNorm(solver,
```

HYPRE 44

Getting the code

<http://www.llnl.gov/CASC/hypre/>



- The User's Manual and Reference Manual can be downloaded directly
- A form must be filled out prior to download. This is just for our own records.

CASC

HYPRE 45

Calling *hypre* from Fortran

- C code:

```
HYPRE_IJMatrix A;
int          nvalues, row, *cols;
double       *values;

HYPRE_IJMatrixSetValues(A, nvalues, row, cols, values);
```

- Corresponding Fortran code:

```
integer*8      A
integer        nvalues, row, cols(MAX_NVALUES)
double precision values(MAX_NVALUES)

call HYPRE_IJMatrixSetValues(A, nvalues, row, cols, values)
```

CASC

HYPRE 47

Building the library

- Usually, *hypre* can be built by typing `configure` followed by `make`
- Configure supports several options (for usage information, type '`configure --help`'):

‘`configure --enable-debug`’ - turn on debugging
‘`configure --with-openmp`’ - use openmp
‘`configure --with-CFLAGS=...`’ - set compiler flags

CASC

HYPRE 46

Reporting bugs

<http://www-casc.llnl.gov/bugs>

- Submit bugs, desired features, and documentation problems, or query the status of previous reports
- First time users must click on “Open a new Bugzilla account” under “User login account management”

CASC

HYPRE 48

This work was performed under the auspices of the U.S.
Department of Energy by Lawrence Livermore National
Laboratory under contract no. W-7405-Eng-48.

CASC

HYPRE 49